ARMY RESEARCH LABORATORY

Real-Time Radio Wave Propagation for Mobile Ad-Hoc Network Emulation and Simulation Using General Purpose Graphics Processing Units (GPGPUs)

by Brian J. Henz, David Richie, Song Jun Park, and Dale R. Shires

ARL-TR-6764 May 2014

NOTICES

Disclaimers

Army Research Laboratory

Aberdeen Proving Ground, MD 21005

ARL-TR-6764 **May 2014**

Real-Time Radio Wave Propagation for Mobile Ad-Hoc Network Emulation and Simulation Using General Purpose Graphics Processing Units (GPGPUs)

Brian J. Henz, Song Jun Park, and Dale R. Shires
Computation and Information Sciences Directorate, ARL

David Richie
Brown Deer Technology, Forest Hill, MD 21050

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
May 2014	Final	June 2011 to June 2013

4. TITLE AND SUBTITLE Real-Time Radio Wave Propagation for Mobile Ad-Hoc Network Emulation and Simulation Using General Purpose Graphics Processing Units (GPGPUs)	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Brian J. Heinz, David Richie, Song Jun Park, and David R. Shires	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: RDRL-CIM-C Aberdeen Proving Ground, MD 21005	8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-6764
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

primary author's email: <brian.j.henz.civ@mail.mil>

14. ABSTRACT

Large scale experimentation and analysis of mobile ad-hoc networks (MANETs) is an expensive and time consuming task. Even with the best planning, the environment at the time of the experiment is unpredictable, making large scale controlled experiments impossible to perform. By simulating the physical medium it is possible to create a repeatable environment. Real-time radio wave propagation path loss calculations are a key component in creating a virtual environment for MANET simulation, emulation and experimentation. There are many algorithms available for computing the radio wave propagation path loss. In this paper we investigate the use of the Longley-Rice model computed using graphics processing units (GPUs). The goal of this effort is to solve the Longley-Rice algorithm in real-time for thousands of transmitters and receivers. We will discuss the choice of the Longley-Rice algorithm, algorithm modification for GPUs, precision issues and optimization. This method will be demonstrated in the context of a dedicated high performance computing system for MANET simulation, emulation and experimentation.

15. SUBJECT TERMS

Mobile Ad-Hoc Network Emulation, General Purpose Graphics Processing Unit, RF Propagation Path loss

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Brian J. Henz
Unclassified	Unclassified	Unclassified	UU	28	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6531

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to acknowledge the support received from the High Performance Computing Modernization Program Office (HPCMPO) under the Mobile Network Modeling Institute (MNMI). The authors would also like to acknowledge the support of the High Performance Computing Modernization Program (HPCMP) High Performance Computing Modernization Program Productivity Enhancement, Technology Transfer and Training (PETTT) program.

1. Introduction

Large scale testing, evaluation and analysis of mobile ad-hoc network (MANET) platforms is an expensive proposal with a limited parameter space and repeatability under experimental conditions (*1*). Therefore, simulation and emulation tools have been developed that provide researchers with a controllable and repeatable environment for analysis of MANET platforms. In particular, emulation holds great promise for limiting the amount of live experimentation required for MANET platform development. Emulation provides for hardware-in-the-loop (HIL) testing and analysis where the physical medium is replaced by a virtual environment and a physical or simulated radio can be used with real applications. Much effort has been performed in this area to make the virtual environment as physically meaningful as possible (*1–3*) but one limitation that remains for real-time emulation is the accurate computation of the radio frequency (RF) propagation path loss between radios.

MANET emulation has traditionally relied on either off-line link analysis including calculations of RF propagation path loss (*2*) or real-time calculations with stochastic models (*3*). When calculations are performed off-line it is assumed that the node mobility is known apriori. This is a severe limitation when experiments may involve live components or mobility is controlled by a third party application such as a force modeling simulation. One method to remove this limitation is to use interpolation between known data points but the accuracy of this method is limited by a number of factors. These factors include the physical size of the virtual environment, machine memory for storing and accessing a look up table, signal phase, and fading affects from small obstructions are not captured because of the computed grid size. Free-space models are not a satisfactory solution either as they do not capture the effect of terrain, vegetation, precipitation, or man-made structures on RF path loss.

The increasing fidelity of MANET emulations from packet-level to signal-level (*4*) analysis will require fast and accurate modeling of the physical layer (*5, 6*). We present here a solution that combines the accuracy of physics-based models for RF propagation with real-time performance. Large scale mobile ad-hoc networks such as those deployed by the U.S. Army may have more than 5000 devices in a given physical space. In order to compute the path loss experienced by these devices in real-time we have chosen to develop the Longley-Rice model for use with Graphics Processing Units (GPUs) (general purpose graphics processing units). The following paper describes the algorithm, GPU development and use within a MANET emulator for the emulation of large scale MANETs.

2. Large Scale MANET Emulation

The primary goal of the MANET modeling effort at the U.S. Army Research Laboratory (ARL) is the development of a framework for large scale MANET emulations, namely up to 5000 emulated devices. A large scale emulation environment provides a testbed for the research, development, and evaluation of network algorithms, applications and devices in a controlled environment (*7*). For instance, it is possible to use the Optimized Link State Routing daemon (OLSRd) ad-hoc wireless mesh routing daemon (*8*) directly from the source code repository or modified for a particular analysis. In addition to the use of unmodified software applications (*9*), this environment allows for the integration of virtual devices into live experiments in order to augment the testing parameters and the perceived traffic by physical network devices in the field. This augmentation of live experiments enhances the experience of testers and increases the degrees of freedom that can be evaluated in an experiment. The achievement of these goals for large scale MANET emulations and live experiment integration requires the bridging of a number of technical gaps. One of the bridging technologies presented here is the development of a real-time RF propagation computation and the hi-fidelity software emulation of multiple MANET waveforms.

Providing an accurately modeled physical medium for RF propagation is a requirement for emulating MANETs with confidence. A worst case link analysis for a MANET scenario includes all point-to-point links with continuously updated RF path loss information. For a 256 transceiver emulation this would require over 65,000 RF path loss calculations per second. An efficient CPU implementation of the Longley-Rice model, as will be discussed later, is about 20 times slower than the GPU implementation on an NVIDIA 2070 device using OpenCL. This performance discrepancy is significant for MANET emulation since the radio models require a general purpose CPU to host a full operating system for execution. The GPU can be hosted on the same commodity motherboard and perform the path loss calculations independently from the emulated network stack.

3. The Longley-Rice Algorithm for Real-Time Radio Wave Propagation Path Loss

RF wave propagation models play an essential role in the planning, analysis and optimization of radio networks (*1, 10–13*). For instance, coverage and interference estimates of network configurations are based on field strength predictions, routing is also highly dependent upon computed path loss data (*1*). Approaches for field strength prediction can be divided into semi-empirical and ray-optical models. For example, the semi-empirical COST-Walfisch-Ikegami model (*14*) estimates the received power predominantly on the basis of frequency and distance to the transmitter. Ray-optical (*15*) approaches identify ray paths through the scene, based on wave guiding effects like reflection and diffraction. Semi-empirical algorithms usually offer fast computation times but suffer from inherent low prediction quality. Ray-optical algorithms feature a higher prediction quality at the cost of higher computation times.

Emulation of MANETs must execute in real-time in order to provide HIL compatibility. At the physical layer, the interactions between devices is governed by the RF propagation characteristics of the environment. It is therefore imperative that the RF path loss data must be computed and provided to the emulation environment in real-time. The algorithms used to compute path loss must be computed in real-time for each of the possible propagation paths. Initially assuming that all devices in a single emulation scenario are within propagation range of each other the computational complexity of the RF path loss algorithm is O(n^2), where n is the number of transmitter/receiver device pairs in the scenario. Multiple methods exist for computing the RF path loss data including both (semi-) empirical and ray-optical models. Both methods require a large number of floating point operations, necessitating a high floating point operations (FLOP) rate for real-time path loss predictions.

Recently, the use of GPUs have been identified as a solution to provide the raw floating point performance (*16*) required to compute the RF propagation path loss in real-time (*12, 13*). Originally, GPUs were developed in order to quickly compute rasterization which requires a large number of simple floating point operations. This targeted design is the reason that the architecture has been able to exceed the performance of CPU architectures for raw FLOP rates (*17*). In addition to the raw floating point performance of the GPUs, the GPUs are tightly coupled with the underlying system including the network interface cards (NIC). High bandwidth communication through the PCI Express interface to the NIC provides a path for wireless device positions to be read in and RF path loss data to be pushed out, thus allowing for a tight coupling between the MANET emulation environment and the RF path loss calculations. The MANET emulation environment used here is Extendable Mobile Ad-hoc Network Emulator (EMANE) from DRS (formerly Cengen Labs) (*18*). In EMANE, the GPS locations of all mobile radios are transmitted over a multicast group that is monitored by the emulated devices for self-location. The RF propagation calculation joins this multicast group and collects all of the radio positions and computes results as fast as possible. Typically the GPS coordinates are transmitted on 1 s intervals. For a 1 s transmit interval, it can be safely assumed that the GPU calculation should take less that 0.5 s in order to allow time for the broadcasting of results to the mobile radios, and the collection of updated positions for a subsequent calculation. The calculation cost for a particular scenario is very consistent as the number of terrain profile points remains constant for each point-to-point calculation, regardless of distance, and the number of radios also remains constant. Under these conditions and the fact that the amount of data copied to and from the network and across the PCI bridge is fairly small, e.g. with 5000 radios we have 960 kb/s of position data and 800 Mb/s of single precision path loss data distributed across 114 GPU host nodes in our host system. Relative to the NIC and PCI Express bandwidths of 10 Gb/s and about 32 Gb/s, respectively, a 0.5 s buffer for these actions is conservative. If, by chance a GPS update is missed because of a delay in processing or communication, then the system will just wait for the next update and continue from that point. This is acceptable because 1 s GPS updates are fairly frequent for path loss considerations in non-urban scenarios.

Although a number of path loss algorithms exist, we down-selected the methods based on the criteria we had for a non-urban environment and a large number of wireless devices. In order to provide a robust path loss calculation for non-urban environments we selected the Longley-Rice model. The Longley-Rice model is capable of predicting path loss in an area or point to point mode, with the later used here. Longley-Rice is designed for frequencies between 20 MHz and 20 GHz and for path lengths between 1 km and 2000 km (*19*), both within our scenario operating ranges. In point-to-point mode the model considers input parameters such as distance, antenna height, surface reflectivity, climate and the terrain profile between the transmitter and receiver (*20*). The rest of the environmental parameters can be transient or fixed upon initialization. This implementation is robust in that it allows all parameters to change each time the GPU kernel is executed.

In order to meet the real-time requirements for RF propagation path loss estimation suitable for network emulation, an implementation of the Longley-Rice (*21*) Irregular Terrain Model (ITM) was developed for GPUs. This implementation is based on the open-source C implementation available from the U.S. Department of Commerce (*22*). The code development required significant re-factoring to employ algorithms suitable for the fine-grained parallelism of modern

GPU architectures. Two algorithms account for a significant portion of the overall execution time, namely the point-to-point path loss calculation and the digital terrain extraction algorithm. Both of these algorithms have been ported to the GPU for acceleration. This allows the entire propagation path loss calculation to be performed on a GPU with the GPS coordinates of each radio as an input, and the path loss matrix as the output.

4. Adapting the Longley-Rice Algorithm for Efficient Computation on Many-Core Architectures

Algorithm development for GPUs and many-core architectures, in general, is a difficult task when attempting to achieve acceptable performance. Algorithms for modern CPUs assume a well managed memory hierarchy with multiple levels of cache to keep the computing units supplied with data for calculations. GPUs and many-core architectures such as the Bulldozer processor from AMD typically have less dedicated memory per compute core and data layouts that must be user-managed in order to keep the computing units from becoming starved for data. In this section we discuss the details of developing the Longley-Rice algorithm for a many-core processor, specifically targeting GPUs.

4.1 Porting the ITM algorithm to GPU

Significant challenges were encountered in the initial development of the GPU algorithm based on the serial CPU algorithm. First, the C code implementation of the serial algorithm closely reflects the original FORTRAN implementation and contained many basic constructs ill-suited to modern processors, which required substantial reformulation. An example of this reformulation was the replacement of nested conditional control flow inside inner loops. Second, the original double-precision algorithm had to be converted to single-precision raising the issue of numerical stability within the complex formulas that comprise the Longley-Rice algorithm. There exists a significant, i.e., greater than 100%, performance disparity between single- and double-precision with modern GPUs, which will likely continue going forward. Further, even though high-end GPUs now provide a substantial double-precision capability, this does not extend to complicated transcendental functions which are pervasive within the Longley-Rice algorithm.

Initial development of the Longley-Rice algorithm focused on the ITM algorithm LRPROP routine. LRPROP was implemented in both the Brook+ and CUDA languages. Subsequently, the OpenCL[1] API was released for the development of architecture independent applications. OpenCL™ is an industry standard programming API for parallel programming of heterogeneous computing platforms (*23*). The use of OpenCL ensures portability across modern multi-core and many-core processors and specifically supports the use of GPUs from AMD and NVIDIA, as well as CPUs from Intel and AMD. ITM was developed for GPUs using the standard compute layer (STDCL) (*24*) API. STDCL leverages OpenCL and presents it in a more efficient and simplified interface designed for the development of complex high performance computing (HPC) applications.

The ITM path loss calculation is implemented using 10 OpenCL kernels as shown in figure 1, including the five kernels that make up LRPROP that were also implemented in the architecture specific Brooke+ and CUDA languages. The kernels are executed in succession using a digital terrain map pre-loaded into the main GPU memory attached to the GPU co-processors. Initialization of the compute cycle begins when and array of transceiver positions is supplied through the network interface, or from a calling library for stand-alone applications. Following the transfer of position data to the GPU, the ITM algorithm is executed. Details of the computation performed and result distribution through the return of a path loss matrix for all transmitter/receiver pairs follows.

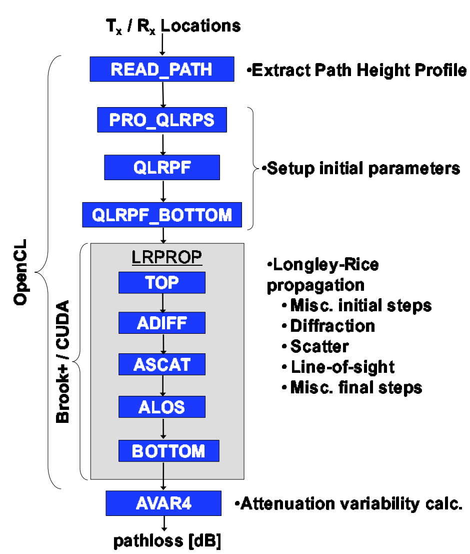

Figure 1. Diagram of ITM routines developed for Brooke+/CUDA and OpenCL.

[1]OpenCL is a trademark of Apply Inc. used by permission by Khronos.

In figure 1 the first kernel started, READ_PATH, is started to extract the path data from the elevation map for each transmitter/receiver pair using the preloaded digital terrain data. The result is a segmented height profile along the path connecting the transmitter and receiver, including a correction for the curvature of the earth. The path data for each transmitter/receiver pair is subsequently stored in an array located in global GPU memory and made available for the next stage of the ITM algorithm. The array of height profiles is pre-processed with three kernels, namely PRO_QLRPS, QLRPF, and QLRPF_BOTTOM that were re-factored from the original ITM C code implementation. The results are multiple arrays of per-path quantities required in the next stages of the computation.

The computational core of the Longley-Rice algorithm in LRPROP, is broken down into sub-calculations for different physical effects that impact the total RF path loss value. The kernels ADIFF, ASCAT, and ALOS implement models for diffraction, scatter, and line-of-sight attenuation, respectively. The final kernel AVAR4 implements the total attenuation and variability computation that results in the final path loss estimate for each transmitter/receiver pair. The result of the computation is a matrix of path loss values generated in stripes across multiple GPUs per node, with each node responsible for a subset of the complete matrix of transmitter/receiver pairs. After computation, the path loss data is distributed to the radio models via multicast messages that contain an array of values for the neighboring nodes. The RF path loss calculations ignore the direction of the antenna and the final RF path loss value must be augmented with antenna gain information based on directional data provided by EMANE.

Efficient distribution of computational load to the GPU is performed by partitioning the point-to-point paths into blocks of predefined size. For currently available hardware and our scenarios, a block size of 4096 paths is used. This distribution method is capable of treating arbitrary numbers of radios by performing computations over multiple blocks and padding the blocks when the number of paths is not commensurate with block size. We have considered two mechanisms for treating systems with multiple GPUs. For OpenCL a multidevice context can be used with the work being distributed and evaluated on multiple GPUs concurrently. Alternatively, the features of STDCL that enable the transparent use of multiple devices can be used to divide the work across independent processes for very a large system. No performance advantage of one method over the other is anticipated as the number of kernels executed on each device and the memory accesses will remain consistent. Since we have access to multiple nodes, each with multiple GPUs, we have focused our attention on the latter method.

4.2 Numerical Stability and Consistency Across Architectures

As noted previously, single precision floating point computations are used on the GPU in order to achieve maximum performance, the trade-off being possibly decreased accuracy. Since the Longley-Rice model uses statistical estimates to compute the variability of signal pathloss due to situation, time and location. The actual received signal is expected to deviate from the computed value due to these variables but the model still provides an reasonable estimate. Therefore, small variations due to single versus double precision are not expected to invalidate the computed results for its intended purpose of estimating signal loss over irregular terrain. For the Longley-Rice algorithm there are a large number of transcendental functions that do not have a double precision computation available. Algorithm development with single-precision accuracy raised concerns with numerical stability and consistency, especially, in the context of forward and inverse transcendental functions with small angles. Whereas it is possible, although not guaranteed, that reasonably precise consistency might be expected across these architectures for simple algorithms based on multiply-add operations, the complexity and reliance upon complex transcendental operations makes exact agreement here unlikely. Factors impacting the difference in results include extended bit precision used in some operations, differences in rounding behavior, and differences in the software implementation of complex operations. Additionally, the GPU implementation introduces the possibility of order-of-operation effects as a result of the fine-grain parallelism within some kernels.

An issue identified across many elements of the algorithm was the repeated use of forward and inverse transcendentals at small angles. An example of this small-angle effect is the use of great circle calculations over small areas in which the correction due to the curvature of the earth was small. A serious numerical instability was identified with the pattern of successive operations of cosine, followed by a minor calculation, and then followed by an arc cosine. Such patterns had the potential to produce an intermediate value slightly greater than 1.0 and a final result of NaN. The effects of this numerical instability can be complicated and the impact on the final path loss can range from a small error to an undefined result (NaN). In some cases a less severe numerical error results from differences in transcendental functions at limiting values. Secondary impacts were also identified, for example differences in the projected map location within the digital terrain map can introduce differences in elevation within the extracted height profile that only impact results by changing the statistical metrics calculated for these height profiles. The solution to many of these issues was to re factor the formulas found in the original reference implementation and introduce forms with greater stability at the limiting ranges found within the typical uses cases. Consider the original distance calculation, that begin by first calculating,

$$\begin{aligned} a = &\cos(90 - lat_2) * \cos(90 - lat_1) + \\ &\sin(90 - lat_2) * \sin(90 - lat_1) * \\ &\cos(lon_2 - lon_1) \end{aligned} \tag{1}$$

Where lat_1, lon_1 refer to the transmitter coordinates and lat_2, lon_2 refer to the receiver coordinates. Using the value a computed in equation 1,

$$b = \arccos(a) \tag{2}$$

Where for the earth,

$$distance = R_{earth} * b \tag{3}$$

Here R_{earth} is the radius of the earth. For small angles this calculation can be unstable using single precision so we used the following approximation,

$$\Delta lon = lon_2 - lon_1 \tag{4a}$$

$$\Delta lat = lat_2 - lat_1 \tag{4b}$$

$$\begin{aligned} a = &(\sin(\Delta lat))^2 + \\ &\cos(lat_1) * \cos(lat_2) * \left(\sin\left(\frac{\Delta lon}{2}\right)\right)^2 \end{aligned} \tag{5}$$

$$b = 2 * \arcsin(\min(1, \sqrt{a})) \tag{6}$$

Distance is then computed using equation 3. Efforts to improve the numerical stability resulted in good agreement between a CPU and AMD Cypress and Cayman GPUs. We take as an assumption that the CPU hardware provides a reasonable baseline for comparison since the implementation of all relevant math operations are well established, more thoroughly tested, and provide better edge cases relative to GPUs. Results for the NVIDIA Fermi GPU exhibited notable discrepancies, with a complete understanding of the cause remaining for further investigation. Numerical consistency was tested across these architectures using a simple synthetic test case involving an 8 by 8 uniform grid of radio transceivers over a digital elevation map (DEM) with 1.2 million elevation points. Table 1 shows the percentage of the point-to-point path loss results calculated on a particular GPU architecture that agree with the results calculated on the CPU to within a tolerance of 1 dB, 2 dB, and 10 dB, respectively.

Table 1. Consistency of the results calculated with various GPUs compared to the baseline results from the CPU.

Processor	<1 dB	<2 dB	<10 dB
ATI Radeon HD 5870	98 %	99 %	100 %
AMD Radeon HD 6970	98 %	99 %	100 %
NVIDIA Tesla C2070	86 %	90 %	94 %

As observed in table 1, the ATI/AMD devices provide a result more consistent with the baseline CPU. We have been unable to determine at this time the cause of the discrepancy between the two vendors but the ATI/AMD solution consistently provided results more consistent with the CPU baseline calculations.

4.3 Performance and Scaling

In this section we explore the achieved Longley-Rice algorithm performance on several GPU platforms, in the process comparing vendor solutions from ATI/AMD and NVIDIA. It should be noted again that the Longley-Rice algorithm is heavily dependent upon transcendental functions and not on more typical multiply add (MADD) operations. This is an important point as the reported FLOP rates are for MADD operations, and transcendental function performance is not directly related. The timing results in table 2 list the wall clock time required for three different architectures to compute all point-to-point RF path loss values using the Longley-Rice algorithm.

Table 2. Timing results for 256 transmitters/receivers using the OpenCL version of the Longley-Rice algorithm run on AMD and NVIDIA GPUs.

Processor	Time (s)
ATI Radeon HD 5870	0.72
AMD Radeon HD 6970	0.55
NVIDIA Tesla C2070	0.39

As illustrated in table 2 using the current ITM implementation, all of the tested GPU architectures are capable of providing computed RF path loss results for 256 transceivers, or 65,536 point-to-point calculations, in less than 1 s on a single GPU device. For 256 radios, the fastest time to solution is reported as 0.39 s using an NVIDIA C2070 (*25*) as compared with 0.72 s and 0.55 s using an ATI Radeon HD 5870 and AMD Radeon HD 6970, respectively (*26*). Complete performance results are plotted in figure 2 for a range of 32 to 1024 transceivers.

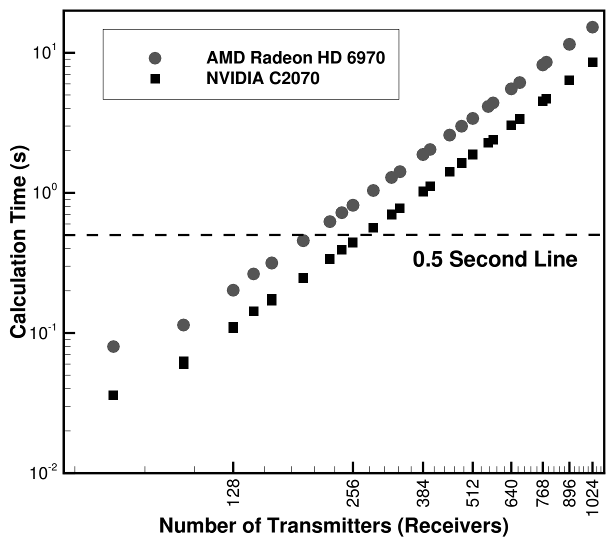

Figure 2. Plot of total ITM (Longley-Rice) calculation time versus number of transmitters/receivers. The 0.5 s line represents the maximum time allowed for real-time computations.

In figure 2 a line is drawn at 0.5 s to show approximately the number of transceivers a particular GPU is capable of considering in realtime. It is interesting to note that the theoretical peak FLOP rate of the AMD Radeon HD 6970 is 2.703 TFLOPs and the NVIDIA C2070 is only 1.288 TFLOPs. Conversely, the number of radios supported by the Longley-Rice algorithm in less than 0.5 s of computation time is higher for the NVIDIA GPU. This apparent inefficiency in the AMD hardware is due to the fact that many of the floating-point operations in the Longley-Rice algorithm are transcendental functions such as cosine, sine, tangent, co secant, etc., The performance of a specific architecture on the Longley-Rice algorithm is therefore not easily predicted by theoretical peak performance. Additionally, the memory access patterns within the kernels are nontrivial, and this will also contribute to the observed performance.

Performance of complex multikernel algorithms can be impacted by many factors including pure computational load, memory access, host-device data transfer, and kernel launch latency. In the case of the 10 kernels in the ITM implementation, each individual kernel shows a very low execution time when directly measured in a fully blocking mode of operation. In order to investigate whether the ITM implementation is effectively using the GPU compute capability, the stripe size over which the computation is distributed was varied to observe the effect of changing the amount of work performed per kernel execution. Initially the stripe size was set at 4096 with subsequent test cases of 2048 and 1024 point-to-point calculations. The results in table 3 show an improvement on the order of 10% when increasing the block size from 1024 to 4096, thus providing more work per kernel execution. This indicates that the block size of 4096 is performing only slightly better than the block size of 2048, therefore increasing the block size further would yield diminishing returns. Increasing the block size further would also decrease the efficiency of performing calculations where the number of point-to-point paths was not commensurate with block size. For example, with a workload of 65,536 point-to-point calculations, increasing the block size will approach the size of the work load resulting in an efficient calculation when the work load is not a multiple of the block size.

Table 3. Performance for AMD and NVIDIA GPUs as a function of block size in terms of the number of point-to-point paths evaluated per kernel execution. Results show that using a block size of 4096 provides sufficient work per kernel execution for efficient use of the GPU.

Processor	Block Size	Time (s)
ATI Radeon HD 5870	1024	83
ATI Radeon HD 5870	2048	75
ATI Radeon HD 5870	4096	72
AMD Radeon HD 6970	1024	65
AMD Radeon HD 6970	2048	58
AMD Radeon HD 6970	4096	55
NVIDIA Tesla C2070	1024	42
NVIDIA Tesla C2070	2048	40
NVIDIA Tesla C2070	4096	39

Intra-node and inter-node scalability for small numbers of GPUs was evaluated here under several configurations using available resources that included a system with four ATI Radeon HD 4870X2 graphics cards with a total of eight GPUs and another system with an ATI Radeon HD 5870 and 5970 with a total of three GPUs. Note that with the dual GPU graphics cards, a PCIe interface is shared between pairs of devices. The scalability results are shown in figure 3 and show reasonable scalability using a multidevice OpenCL context. Scalability to very large systems is being investigated on a large 456-GPU cluster and will be reported elsewhere.

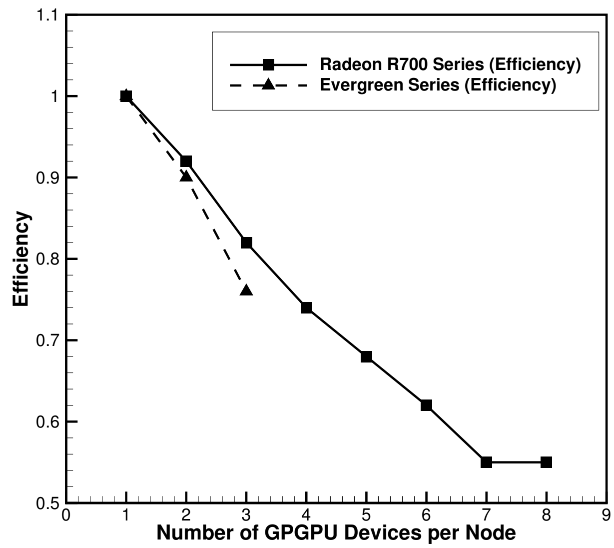

Figure 3. Efficiency of ITM algorithm OpenCL implementation on multiple AMD/ATI GPU device families.

4.4 GPU Limitations

When using GPUs for a complex computational task, a number of limitations must be observed. For instance, the Longley-Rice algorithm includes a high proportion of transcendental computations. GPUs are designed to perform MADD operations very efficiently, whereas the number of computational units on ATI/AMD devices dedicated to complex mathematical functions is approximately 20%–25% of the GPU capacity. In addition the memory access patterns are important because the ratio of memory access to computation cost is much higher for a general CPU architecture. A third factor important for this effort is the number and work load of computational kernels. The final factor considered is the language choice that affects performance and portability, table 4.

Table 4. Performance comparison for LRPROP, the main compute kernel in ITM, using Brook+ and OpenCL.

Architecture	Language	Time (s)
ATI Radeon HD 4870	Brook+	0.02
ATI Radeon HD 5870	Brook+	0.01
ATI Radeon HD 4870	OpenCL	3.50
ATI Radeon HD 5870	OpenCL	0.40

5. Conclusions and Future Work

Using GPUs and a portable standard for parallel computing systems, namely OpenCL, it has been possible to develop a Longley-Rice ITM implementation for real-time RF path loss computations that support MANET emulation. This has in turn provided ARL with the capability to augment live exercises, integrate MANET emulation with force model simulations and to drive programmable attenuators for laboratory experimentation with physical devices. Prior to the development of these capabilities with GPUs the wireless node mobility and path loss needed to either be computed apriori or use a large number of (i.e., 10,000) CPU cores, dedicated to path loss calculation.

A number of issues have been overcome including the use of reduced precision, through the use of alternative calculations for edge cases, and the issue of load distribution and communication costs through the creation of computation blocks that limit kernel calls and minimize wasted computation cycles. The resolution of these issues enables the emulation framework at ARL to provide real-time situational awareness data to live field exercises and will have applicability to the integration with future force modeling simulations. Continuing developments include improving predictions through the addition of foliage effects (*27, 28*) and ITM area calculations.

6. References

1. Schmitz A.; Wenig, M. The Effect of the Radio Wave Propagation Model in mobile Ad Hoc Networks. In *The 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2006)*, October 2006.

2. Kaplan, M. A.; Chen, T.; Fecko, M. A.; Gurung, P.; Hokelek, I.; Samtani, S.; Wong, L.; Patel, M.; Staikos, A.; Greear, B. Realistic Wireless Emulation for Performance Evaluation of Tactical MANET Protocols. In *IEEE Military Communications Conference (MILCOM)*, October 2009.

3. Nitsche, T.; Fuhrmann, T. A Tool for Raytracing Based Radio Channel Simulation. In *SIMUTools* March 2010.

4. Andelfinger, P.; Mittag, J.; Hartenstein, H. GPU-based Architectures and Their Benefit for Accurate and Efficient Wireless Network Simulations. In *IEEE MASCOTS*, July 2011.

5. Eltahir, I. The Impact of Different Radio Propagation Models for Mobile Ad hoc NETworks (MANET) in Urban Area Environment. In *Wireless Broadband and Ultra Wideband Communications, 2007. AusWireless 2007. The 2nd International Conference on*, August 2007.

6. Stepanov, I.; Rothermel, K. On The Impact of a More Realistic Physical Layer on MANET Simulations Results. *Ad Hoc Networks* **2008**, *6* (1), 61–78.

7. Grøtli, E. I.; Johansen, T. A. Path Planning for UAVs Under Communication Constraints Using SPLAT! and MILP. *J. Intell. Robotics Syst.* **2012**, *65* (1–4), 265–282.

8. Tønnesen, A. Impementing and Extending the Optimized Link State Routing Protocol. University of Oslo, 2004.

9. Henz, B. J.; Parker, T.; Richie, D. A.; Marvel, L. Large Scale MANET Emulations Using U.S. Army Waveforms with Application: VoIP. In *IEEE Military Communications Conference (MILCOM)*, November 2011.

10. Rick, T.; Mathar, R. Fast Edge-Diffraction-Based Radio Wave Propagation Model for Graphics Hardware *Antennas, 2007. INICA '07. 2nd International ITG Conference on*, March 2007.

11. Judd, G.; Steenkiste, P. Design and Implementation of an RF Front End for Physical Layer Wireless Network Emulation. In *IEEE 65th Vehicular Technology Conference (VTC2007)*, April 2007.

12. Catrein, D.; Reyer, M.; Rick, T. Accelerating Radio Wave Propagation Predictions by Implementation on Graphics Hardware. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, April 2007.

13. Michéa, D.; Komatitsch, D. Accelerating a Three-Dimensional Finite-Difference Wave Propagation Code Using GPU Graphics Cards. *Geophysical Journal International* **2010**, *182* (1), 389–402.

14. Damasso, E., Ed. *Digital Mobile Radio Towards Future Generation Systems*; Office for Official Publications of the European Communities: Luxembourg, 1999.

15. Bertoni, H. L. *Radio Propagation for Modern Wireless Systems*; Prentice Hall Professional Technical Reference: 1999.

16. Song, Y.; Akoglu, A. Parallel Implementation of the Irregular Terrain Model (ITM) for Radio Transmission Loss Prediction Using GPU and Cell BE Processors. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 1276–1283.

17. NVIDIA CUDA C Programming Guide, Version 4.0, 2011.

18. Various, EMANE Developer Manual 0.7.3. DRS CenGen, LLC, 2012.

19. Hufford, G.; Longley, A.; Kissick, W. *A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode*, 82-100; National Telecommunications and Information Administration, April 1982.

20. Hufford, G. The ITS Irregular Terrain Model, version 1.2.2 The Algorithm. National Telecommunications and Information Administration Institute for Telecommunication Sciences, 1995.

21. Longley, A. G.; Rice, P. L. *Prediction of Tropospheric Radio Transmission Loss over Irregular Terrain, A Computer Method-1968*; Environmental Sciences Services Administration: Boulder, CO, June 1968.

22. U.S. Department of Commerce, Irregular Terrain Model (ITM) (Longley-Rice). `http://flattop.its.bldrdoc.gov/itm.html`, [Online; accessed 26-April-2011].

23. OpenCL Overview. `http://www.khronos.org/opencl/`, [Online; accessed 14-November-2012].

24. Technology, B. COPRTH. `http://www.browndeertechnology.com/coprthr_stdcl.htm`, [Online; accessed 14-November-2012].

25. Benchmarks used the OpenCL implementation provided by the NVIDIA CUDA Toolkit v3.2.

26. Benchmarks used the OpenCL implementation provided by the AMD ATI Stream SDK v2.3.

27. Wang, F.; Sarabandi, K. A Physics-Based Statistical Model for Wave Propagation Through Foliage. *Antennas and Propagation, IEEE Transactions on* **2007**, *55* (3), 958–968.

28. Weissberger, M. A. *An Initial Critical Summary of Models for Predicting the Attenuation of Radio Waves by Trees*; ESD-TR-81-101; Department of Defense Electromagnetic Compatibility Analysis Center, July 1982.

List of Symbols, Abbreviations, and Acronyms

ARL	U.S. Army Research Laboratory
CL	computer layer
DEM	digital elevation map
EMANE	Extendable Mobile Ad-hoc Network Emulator
FLOP	floating point operation
GPGPU	General Purpose Graphic Processing Unit
GPU	graphic processing unit
HIL	hardware-in-the-loop
HPC	high performance computing
HPCMP	High Performance Computing Modernization Program
HPCMPO	High Performance Computing Modernization Program Office
ITM	Irregular Terrain Model
MADD	multiply add
MANET	mobile ad-hoc network
MNMI	Mobile Network Modeling Institute
NIC	network interface cards
OLSRd	Optimized Link State Routing daemon
RF	radio frequency
STDCL	standard compute layer

NO. OF COPIES	ORGANIZATION
1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIM C BRIAN HEINZ

INTENTIONALLY LEFT BLANK.